Spring 4-1-2011

Class Final Grading System

Daniel Jones
Dakota State University

CLASS FINAL GRADING SYSTEM

A graduate project submitted to Dakota State University in partial fulfillment of the requirements for the degree of

Master of Science

in

Information Systems

April, 2011

By
Daniel Jones

Project Committee:

Dr. Stephen Krebsbach
Dr. Ronghua Shan
Dr. Mark Moran

PROJECT APPROVAL FORM

We certify that we have read this project and that, in our opinion, it is satisfactory in scope and quality as a project for the degree of Master of Science in Information Systems.

Student Name: Daniel Jones

Master's Project Title: Class Final Grading System

Faculty supervisor: _____ Date: 4/27/11

Committee member: Kongkun Shan Date: 4/27/11

Committee member: Mark Meraa Date: 4/27/11

ACKNOWLEDGMENT

I would like to extend my appreciation to my coworkers and friends at Daytona State College that have helped me with minute details of this project. Their support and assistance has been invaluable to the successful completion of this project. I'd especially like to thank John Hardebeck, who has been my project partner on countless projects and one of the best friends I could hope to have.

I also need to express my sincere love and appreciation to my loving wife Lori, who has supported me through the best and worst of times. My wife has provided the support I desperately needed to get through the trying times in my life and throughout the MSIS program and I could never find a more caring and supportive individual.

ABSTRACT

This project will be focused on the design and implementation of a Class Final Grading System. The project will be undertaken at Daytona State College as a reimplementation and upgrade of a decade-old Class Final Grading System. The goal will be to implement a large assortment of requested features, in addition to upgrading the code to more modern standards and design.

The original Class Final Grading System was written approximately ten years ago to allow Faculty to input grades for students via a web form. These grades were then stored within Daytona State's ERP database. Unfortunately, once the System was written and all initial bugs were fixed, any further upgrades were abandoned in favor of other projects. This resulted in a system that, while still usable, did not keep up with newer web standards or the ERP's continued upgrade and expansion. Additionally, state and federal regulations regarding grading and grade auditing were not properly dealt with, which resulted in several audit criticisms by the Southern Association of Colleges and Schools (SACS).

Since the issue has expanded beyond simple web standards and now directly relates to the accreditation of the College, the Class Final Grading System will be upgraded and expanded to meet all modern standards as well as implement features that have been requested over the past several years. For instance, one requested feature would be the support of a Shared Grade. This would "copy" the grade for a lecture course to the lab course the student is required to be registered in, since the labs and lectures typically operate as a single course, but do not necessarily have the same instructor. Another requested feature

would be the implementation of last attendance date tracking. This would track a student's last date of attendance if they receive an F for non-attendance.

These features, along with appropriate error checking, form validation, and a more modern look and functionality will be deployed in a two-step process. The first step will involve the design and implementation of the system for College Credit students. This constituent group comprises approximately seventy percent of Daytona State's enrollment. Once the system has been deployed into a production environment and all potential bugs have been resolved, the system may be expanded to include the Adult Education program and other systems that utilize a similar codebase.

The initial design and planning of the system will be during the Summer 2010 semester. The goal will be to have the scope definition and planning done prior to the start of the Fall 2010 semester. During the Fall 2010 semester, the system will be developed and implemented for the College Credit students during the first subsession, or half, of the semester. Once potential bugs are identified and resolved, the full implementation will take place so that the entire system in completely in place by the end of the Fall 2010 semester.

DECLARATION

I hereby certify that this project constitutes my own product, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I declare that the project describes original work that has not previously been presented for the award of any other degree of any institution.

Signed,

Daniel Jones

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

INTRODUCTION

Background of the Problem

The purpose of this project is to re-implement a Class Final Grading System within Daytona State College that conforms to modern web standards as well as state and federal regulations. The current Class Final Grading System was initially designed and implemented in the fall 2000 semester with the goal of allowing for web-based submission of grades by Faculty members. Upon submission of the grades, the student would then be able to immediately view their grades without first waiting for approval of the grade submission by the Records Department. This would reduce the workload on the Records Department, as they would not need to review each grade submission, as well as enable the Faculty member to submit grades electronically. The student would also benefit from the ability to view their grades instantaneously upon the Faculty's submission, which was previously unavailable.

Unfortunately, the initial design and implementation of the system has been sparsely maintained, with a primary focus on matching upgrades to the underlying database table structure. Further upgrades, in the form of requested features or government-mandated alterations, have not been performed for various reasons. Consequently, the system has come under criticism from the Southern Association of Colleges and Schools, or SACS, for its lack of conformance.

In addition to the criticism by SACS, the data entry itself is prone to numerous problems. The drop-down box containing the list of possible grades is prone to use of the scroll wheel, which can drastically alter the intended submission of the grade. Contact information for the student is provided within the class roster, but does not take into account a change in contact information stored within the database. Upon submission, if an error of any

sort is encountered, the submission of the grades stops at that particular student. This requires a service call where a web developer must research the nature of the problem, issue a fix or contact the appropriate department to perform an update to the student's record before finally contacting the Faculty to have them complete submission. This process, as one can imagine, can stretch from hours to days as the problem is resolved, with the students waiting for their final grade submission.

To further compound the issue, upcoming federal and state financial aid regulations will require additional information during grade submission for students that fail a course due to non-attendance. In addition, several feature requests for the system have been received over the past several years with goals to improve data accuracy, prevent erroneous submissions of grades, and reduce potential issues from lack of information for students given an incomplete. There have also been several requests to integrate a system for sharing grades across lectures and labs for previously combined courses. While state regulations have forced Daytona State College to separate lectures and labs into distinctly different courses, the grades are equivalent between these courses, and thus currently require duplicate entry. This, of course, produces the possibility of erroneous submission of grades, which may not be caught by either the Faculty member or the student.

Finally, the design of the current system lacks necessary components to conform to modern web standards and ADA regulations. As a State College, DSC must abide by ADA standards for students, Faculty, and staff. Unfortunately, the web-enabled technologies at DSC have been slow to adopt these principles. The class final grading system is one such example, which is unusable by users with sight difficulties. Furthermore, the HTML code utilized by the system is outdated and deprecated, with some elements not supported by the

modern browsers. This further complicates both viewing and submitting grades, especially for users that are not using legacy software for their grade submission.

Statement of the problem

The current class final grading system in use at Daytona State College, or DSC, is unusable. During submission of grades, a multitude of errors may occur that can prevent proper submission of grades for the students. While the errors within the system are usually data-related, and thus require a resolution by entities other than Software Development, the lack of a true error reporting system requires that each individual service call related to the system be painstakingly investigated.

Furthermore, the system as it exists currently violates several web standards as well as being non-ADA compliant. As an educational institution, DSC is required to meet or exceed ADA regulations. The lack of support for web standards also prevents the system from being utilized on modern browsers, necessitating the use of outdated and non-supported software systems.

There have also been numerous requests to resolve problems within the system as well as implement new features to aid student retention and meet federal and state regulatory standards. Those requests that are specifically tied to state or federal regulations are incapable of being implemented in the current system, as the system was not built with modularity in mind.

Objectives of the project

At a fundamental level, the objectives of the project are to improve the accuracy of grade submission while also implementing all of the government-mandated alterations that

have produced SACS criticisms. Once these fundamental requirements are met, further implementation of requested features will be added before the system is released for vigorous testing and evaluation. The testing of the system will be broken into distinct phases, with the initial focus on the largest, and least complex, constituent group of college credit Faculty. Once a determination is made that the system is working as intended, the system will be expanded to include the adult education Faculty. This will provide ample time to resolve issues found with the system prior to submission of end of term grades.

First and foremost, the system will be designed to meet SACS requirements, as well as state and federal regulations. This will include the aforementioned submission of the last date of attendance for students receiving a failure due to non-attendance. In addition, there will be a vital and necessary change to the system to support all aspects of the ADA. This upgrade will allow sight-impaired Faculty to properly utilize the system without requiring assistance.

Once the basic mandated requirements have been met, additional features that have been requested over time will be implemented. For instance, the shared grading system will be implemented to perform a copy of a lecture grade to the appropriate lab grade of a student. There will also be a redesign to the drop-down box of grades to prevent scroll issues changing the grades. For Faculty submitting an incomplete grade, there will be an ability to state the required materials for completion of the course and assignment of a grade. The student's contact information display will be updated, with the ability to view more contact methods than previously offered. The design of the system will be tailored to match the modern design of college applications. Extensive error checking and submission validation will be deployed to further prevent erroneous data entry and correct potential errors prior to submission of grades.

In addition to the tangible objectives for the new system, there are several objectives that are far less tangible. Chief among these intangible objectives is the reduction of service calls for problems that could be solved in advance of the submission. This would reduce man hours in research and resolution of potential problems dramatically, freeing multiple developers for other projects. Another intangible benefit would be a greater assurance for students that their grades would be accurate and posted on-time.

The final deliverables for the system will be a web-enabled application that provides complete and accurate submission of student grades. This system will provide extensive error checking reporting, as well as proper audit trails to meet SACS and government requirements. The system will also utilize standardized design and code implementations, as well as database tables recently designed and developed to support the extensiveness of our grading system. There will also be support for modern web standards as well as ADA compliance to finally provide the sight-impaired Faculty the ability to utilize the grading system. In addition to the application itself, there will be extensive help documentation created, as well as maintenance and support information for use by web development staff.

LITERATURE REVIEW

Grade submission is a complex problem with a seemingly simple resolution. Regardless of the grade system selected, a faculty member will enter the grades in, validate their selections, and then the grades will be processed into the database or data system that houses all grade values. Students can then access their grades at their leisure via various methods before moving onward in their academic career. Why, then, is grade submission such a complex problem? The devil lies in the details, which in the case of Daytona State College is the rampant customizations beyond what a prototypical solution would offer. Consequently, an off-the-shelf system would require modification to match the specific needs of the institution, thereby questioning its acquisition.

Outside of Daytona State College, many organizations utilize a variety of grading systems for submission of grades. Some of these systems are inherent to the Enterprise Resource Platform, or ERP, in use by the institution. For instance, SunGard Higher Education offers the Banner and PowerCAMPUS products, which have a multitude of services including an online grade submission and systems to allow students to view their grades online (SunGard Higher Education, 2011). Such systems are designed to interact seamlessly with other aspects of the ERP system, and thus avoiding any potential data atrocities that may impact students.

Outside of ERP platforms, there are a variety of systems that provide grade entry and grade management services. These systems range from the simplistic to the complex, but essentially perform the same basic task of grade entry and management. An analysis of these

systems will reveal that they utilize similar structures for storing and retrieving grade data, utilizing either a defined standard or proprietary format. While this may work for institutions that have little customization outside of these cookie-cutter design styles, advanced customization is usually not available.

One would wonder if a customized system that exists as an independently-developed product would satisfy the needs of this project. Indeed, there are several customizable software solutions for grading that are available. For instance, Engrade is a freely-available, completely customizable grading solution that offers features beyond grading such as Discussions, Wikis, Flashcards, and Attendance (Engrade, 2011). Such a system would ideally be able to satisfy most grading requirements, including several of the user requirements that are the very basis of this project.

In addition to already-existent products, there are new ideas for grading and assessment being actively developed to expand the breadth of solutions offered. Some ideas that have been put forth are simply re-hashing of already existent grading solutions, simply rebranded with a fresh look and new price tag. Others, such as AutoGradeMe, take automatic grading to a new level by looking beyond simplistic right and wrong to weighted evaluations of structured responses. For instance, AutoGradeMe automatically checks the "style, language feature usage, code complexity, and design consistency" (Zimmerman, Kiniry, & Fairmichael, 2011) of programming assignments to assist in assigning student grades.

The unfortunate nature of these grading systems, however, is that individual customizations to the level of what is required for use by Daytona State College cannot be easily met. While even the Jenzabar CX ERP already in place provides most of the requirements for grade submissions and management, the additional user requirements that

have been defined exceed those offered by the CX product, as well as most products within the current market.

SYSTEM DESIGN

The first phase of the rewrite for the class final grading system will be to analyze the requirements for the system. The phases will follow the basic format of the waterfall method for system design. These phases will include requirements specification, design, implementation, verification and testing, and maintenance.

Preliminary Information

In order to achieve the objectives, the grading system will need to undergo an entire rewrite, including a complete overhaul of the core operation of the system. With the lack of maintenance performed on the system through the years, the queries and code currently in use will provide only a very small base point from which to re-develop the system. The queries will require a complete re-write, with a focus on data accuracy and speed. Furthermore, the code will need to be re-developed to utilize classes introduced over the past few years, with an additional focus on ease of maintenance and speed of the application.

To that end, the first step will be to define the scope of the project. This scope will define the exact purposes of the project, as well as the breadth of work to be performed on the system. Once the scope has been developed, approval and support for the project must be obtained from Academic Affairs, along with dedicated time within the Software Development schedule to perform the rewrite. Upon completion of the scope's development, an in-depth analysis of the current system must be performed to ascertain the usefulness of the existing system's code and queries, along with the current state of problems with the system. This

analysis will also include a review of support for the system, by both Software Development and Academic Affairs, to ascertain if a replacement system will be deemed politically acceptable.

Requirements Specification

Specification of the requirements is an essential piece for proper implementation of the system while simultaneously discouraging alterations to the scope of the project, known as scope creep. In the case of the grading system, the requirements for the system must be defined at two levels. The first level is the user level, where the requirements of the user will be ascertained, defined, and documented. The second level is the technical level, where requirements will be defined for technical aspects of the system.

The user requirement specification will be handled through a system known within Daytona State College as the Refocus on Innovation, or ROI. This system provides the users with an avenue to submit requests to Software Development for potential implementation, pending approval by the Director of Software Development, Chief Information Officer, and the various Vice Presidents of each constituent area. Potential ROIs that are not selected in a particular ROI cycle, which is roughly a calendar year, can be queued for future implementation if time permits after the end of the ROI cycle or if the ROI can be coupled with other requests to form a combined ROI.

The technical requirement specification will be completed through an evaluation of the accepted user requirements and a thorough review of the current system. This review will cover all aspects of the system implementation, from the underlying code to the user interface design. Additionally, there will be a comprehensive review of the current database structure

for the system already in place, as well as all requested features, to ensure reusability of existent structures whenever possible.

Design

Completion of the requirements analysis will then trigger the design for the system. The system's design will be centered on a user-friendly interface with multiple checks-and-balances to prevent erroneous grade submissions that have plagued the system in the past. Furthermore, the system's design will take into account emergent and existent web and accessibility standards that have been ignored up to now.

Utilizing the user-submitted requirements, a set of requested features was developed to provide a quick reference of the deliverables the users were expecting. This list of features can be seen in Table 1. These requirements were selected from the current and previous ROI cycles and were determined to be most apropos for inclusion in the rewrite of the class final grading system.

Table 1. User Requirements

User Requirements
Require entry of materials required to change an incomplete (I) grade and finish the course.
Allow submission of grades to continue even if a particular student's submission encounters an error.
Require submission of a last date of attendance for students receiving an F for non-attendance.
Provide email verification of successful and failed grade submissions.
Provide a printable schedule and class roster for instructors.

Implement shared grading system to share the lecture grade with the lab grade.

Provide more communication options for reaching students.

After the user requirements were defined, a set of technical requirements were also defined to highlight what Software Development felt was essential to include in part of the rewrite of the system. Most of these requirements are centered on systems that have been in place and utilized within other systems at DSC. Consequently, the addition and implementation of these systems could be completed with little additional investment in time, given already-existent frameworks.

Table 2. Technical Requirements

Technical Requirements
Implement W3C and ADA compliant HTML.
Implement the jQuery framework, and DSC-specific CSS styles and JS frameworks.
Implement the DSC Web Help system.
Utilize the DSC utilities classes.
Implement AJAX panels for supplemental information.
Rewrite SQL statements to utilize appropriate fields and indexes, while also reducing duplicated information.
Implement support for table-driven informational text.
Utilize proactive error checking to prevent submission of grades for students with known error conditions.

The basic system will flow through a series of 4 separate and distinct pages of code; each specialized to perform a very specific task or series of tasks. This flow can be seen in

Figure 1. As one will notice, the basic flow of the system is designed to be streamlined and simplistic, despite the complexity of the underlying code. Despite this, each individual page will have a multitude of sub-processes that must be executed and complete successfully to allow grading of a course to continue.

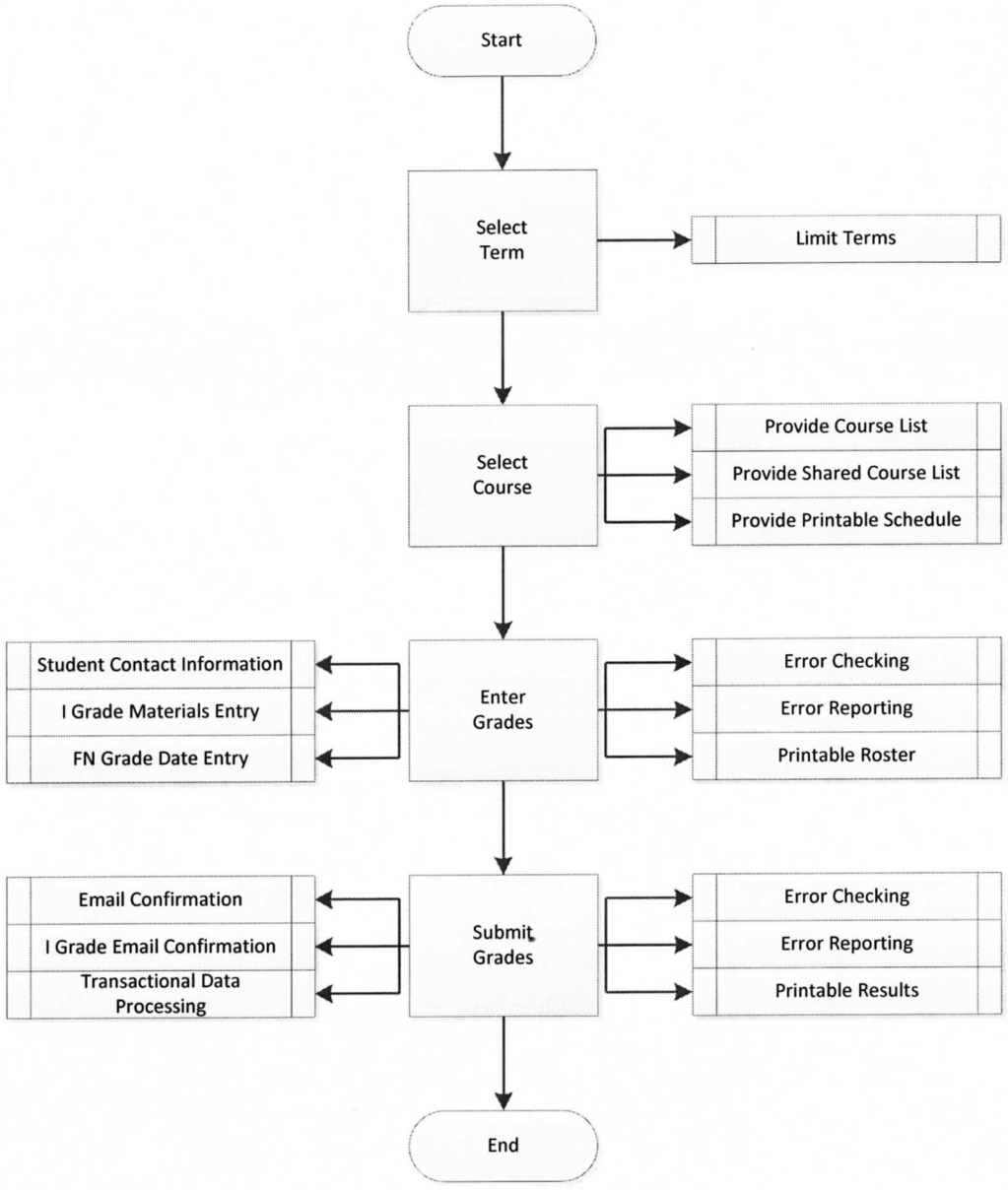

Figure 1. Class Final Grading System Flowchart

Implementation

While individual pieces of implementation may be small and require only small investments of time, the schedule does not permit for dedicated development of a system at this time. Consequently, the project plan must take into account breaks from development to work on other projects and other unforeseen obstacles.

The first step in implementation is to implement the requested database changes. This includes support for the shared grading system, last date of attendance for FN (failure due to non-attendance) grades, I grade support within the TRACS system, and modification of the Catalog Maintenance system to support the shared grading system. The TRACS system has been utilized by DSC Faculty in the past for indicating students at risk of failure, and provides a means to store the text of I grade submissions without excessive additional development. Furthermore, it will provide an additional level of data access and retention, as Academic Advisors assigned to the students can view the required material list should the Faculty or Department Chair be unable to do so. This may also provide for the ability to Advisors to assist students in returning the appropriate materials to convert their grade into a passing grade, thereby promoting student retention and success.

After the database changes are complete, the term selection page rewrite will proceed. To provide for easy testing during development, a system will be devised to quickly switch the logged in Faculty, allowing for impersonation of the Faculty member for debugging purposes. Furthermore, manual data entry for the term selection will be removed in favor of a semester-sensitive selection box limited to a certain range of semesters.

Once the term selection page has been rewritten, work will proceed on the class list page. This will require development of a new query to identify the sections properly and resolve problems with cross-listed courses that have resulted in erroneous displays in the past.

Excessive meeting information will be placed into an AJAX information panel, utilizing a jQuery tooltip component. Those courses that are part of the shared grading system will be placed into a separate display, without the ability to view the class roster or submit grade errantly.

Completion of the class list page will trigger the rewrite of the student list page. This page will contain the greatest breadth of changes, and will require writing numerous queries to ensure student records are accurately recorded. Consequently, there will be a significant investment of time in development of the queries for verifying student records, as well error reporting for these queries. There will also be a significant investment of time in support of incomplete grade submissions and the F for non-attendance system. Once both of these requirements are met, most of the remaining time will be spent on data validation and prevention of submission. A summarized view of the coursework verification and validation procedures can be seen in Figure 2.

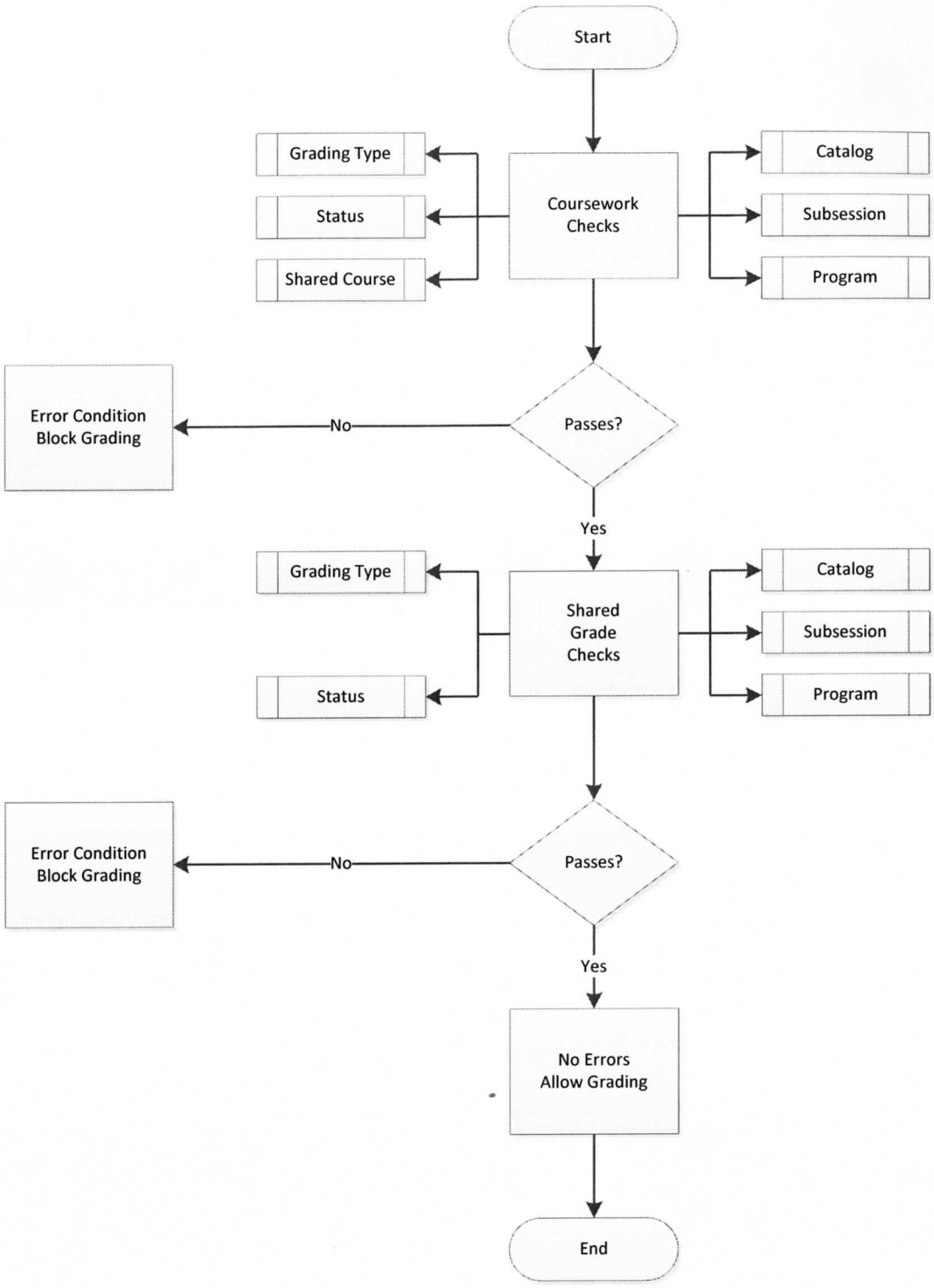

Figure 2. Coursework Validation Flowchart

The final page to receive a complete rewrite will be the actual grade submission page. Much like the previous page, a significant amount of time will be spent on implementation of

requested systems such as the incomplete grade submission and F for non-attendance. Further time will be spent on development of the actual insertion and update queries for regular student records, as well as extensive error reporting. Finally, this system will be the first web-enabled system to utilize proper transaction controls within a SQL database. This will prevent submission of a particular record until it is determined that all requested queries executed without error. A process flowchart detailing this verification and submission of student grades can be seen in Figure 3.

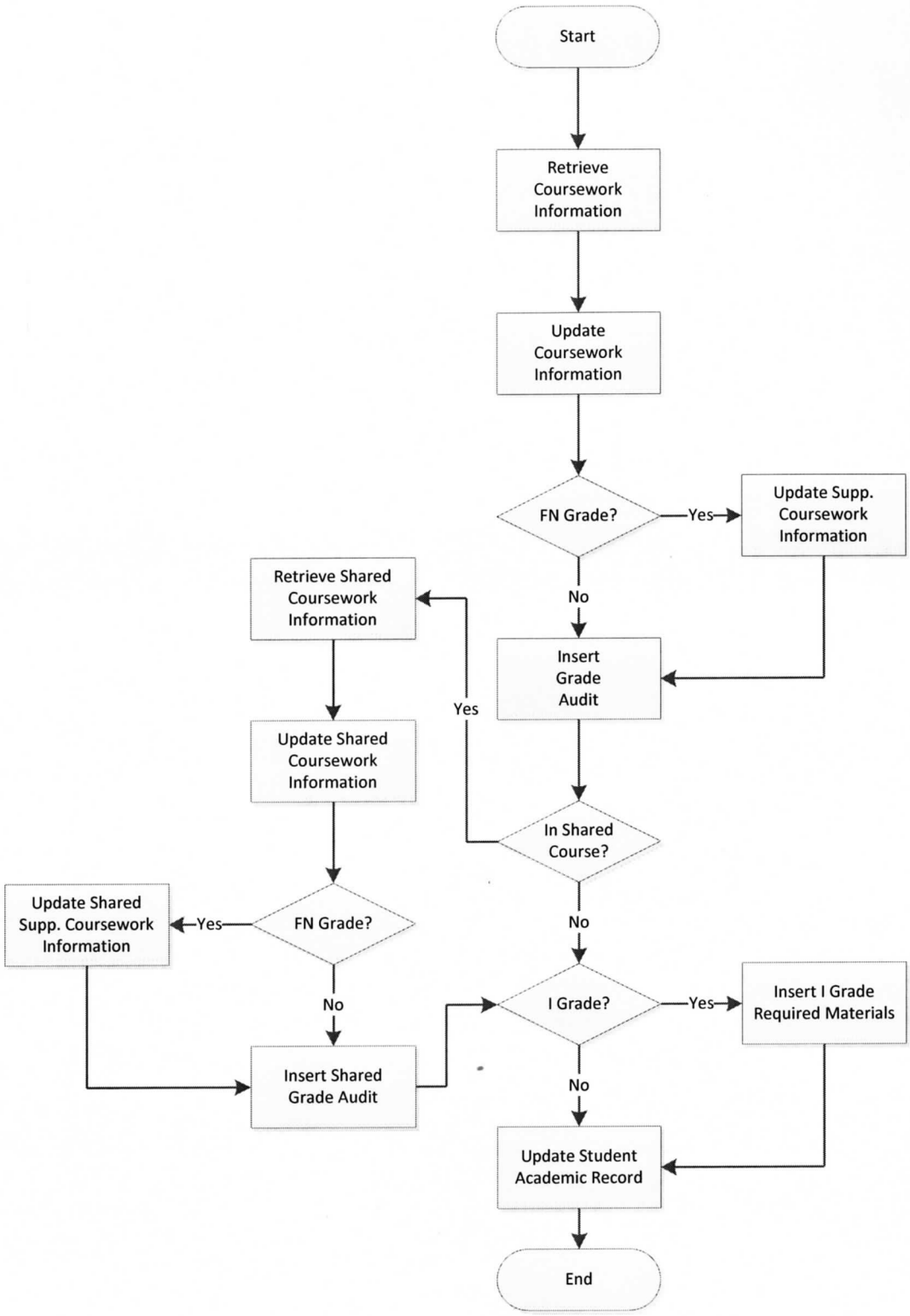

Figure 3. Grade Submission Flowchart

With the entire system completely done, time will also be invested in a new page to offer the instructor a printable schedule. The previous system lacked a printable version of the schedule for the instructors to utilize, necessitating a manual copy-and-paste procedure into Word. This was further complicated by either lack of knowledge of Word by the Faculty member or a schedule structure that was so unusable that modification within Word for print capability was highly improbable.

During the development of all pages, there are several steps that will be universally performed. This includes updates to utilize class architectures that have been recently developed for repetitive tasks as well as an updated design that was pre-made for all of our web apps some time ago. Furthermore, there will be added support for new CSS stylesheets and JavaScript frameworks, such as jQuery, and ADA/W3C-compliant HTML. Finally, each page will utilize the Web Help system, which provides full-page and context-sensitive help for users of enabled systems. This was designed expressly to limit routine service calls and allow the users to help themselves whenever possible.

Once development has been completed, the system will move into an internal testing phase utilizing test Faculty and courses. The test Faculty members will be added into the aforementioned Faculty impersonation system and testing will proceed according to use cases developed on previous grade submissions. As bugs are found during the test phase, the necessary fixes will be implemented and the use case re-run to ensure proper operation. Once all use cases test successfully, the initial testing phase will be complete.

Verification and Testing

With initial testing complete, the system will move into beta testing utilizing Faculty from a wide range of subjects. This list will include Faculty identified for their technical

knowledge, or even lack thereof, as well as previous grading submission problems. The list will also include Faculty with no previous experience submitting grades, as well as Faculty with no previous record of erroneous submissions. Once the list of Faculty is established, approval will be sought, and ideally obtained, from the appropriate Department Chairs before testing proceeds. During the testing phase, bugs found within the system will be resolved, and test data purged to allow re-submission of the data by the appropriate Faculty member. When all submissions are considered successful, beta testing will be complete, and the system will be prepared for final demonstration and implementation.

Prior to final implementation of the system, a demonstration must be arranged for Department Chairs and Academic VPs. This will include preparation of use cases for common grade submissions, as well as specifically-introduced errors to highlight the system's ability to catch and notify the appropriate parties of errors in student records or submission. Any changes requested during this demonstration will be performed after the demonstration and then submitted for final approval by Academic Affairs. With approval of Academic Affairs, the system is considered complete and ready for deployment.

Maintenance

The project will be released prior to exam week within the first subsession of the fall 2010 semester. This will allow time to tweak any particular bugs found during the switch from the test environment to the production environment. Once grading has commenced, time will be dedicated to resolution of bugs found, if any, during the submission process. Should erroneous submissions occur within the system, the Records Department will purge the erroneous submission, allowing time for the code to be fixed before the Faculty attempts a re-submission. Once all bugs are removed and all submissions considered successful, a

project debriefing will be prepared for the Director of Software Development. This debriefing will encompass the entirety of the project plan, along with problems encountered along the way of both a technical and political nature. Any maintenance and support notes for the system will be provided to the Director for dissemination, and final approval of project completion will be obtained. Once the final project approval is received, the project is considered complete, and all further actions are considered maintenance and support.

CASE STUDY (RESULTS AND DISCUSSION)

Implementation of the new grading system went smoothly with only small bugs found and quickly resolved. Overall, the system was met with enthusiastic acceptance, though some users were hesitant with utilizing the systems for tracking incomplete and failure for non-attendance grades. There was also some brief resistance from a particular department that felt excluded from the presentation of the system, though the end result was poor intra-departmental communication.

The most significant benefit was found in the identification and resolution of problems with students before grades were submitted. In previous semesters utilizing the older system design, these students would have caused a blank white screen upon grade submission, with the actual submission process stopping at the student with the erroneous data. Consequently, grades for the entire class would be delayed days or even weeks while the problem was corrected and grades were re-submitted.

Furthermore, service calls to resolve problems with the grading process were greatly reduce. As Table 3 shows, the number of service calls after the system was introduced dropped dramatically. This translated into several tangible and intangible benefits such as increased satisfaction from the faculty members, less stress on the Records staff for manual entry of grades, and less developer time spent on service calls and system maintenance.

Table 3. Grading Service Calls per Semester

Semester	**Number of Service Calls**
Fall 2009	84

Spring 2010	62
Summer 2010	33
Fall A, 2010	4
Fall B, 2010	2
Fall 2010 (total)	7

The use of the incomplete grade submission tracking also reduced the overall number of incomplete grades awarded while also providing Department Chairs with a record of the required materials for switching the incomplete grade to a proper final grade. As one can see from Table 4, the requirement to fully document incomplete grade submissions resulted in a drastic decline in improperly-awarded incomplete grades. The introduction of this system proved to be fortuitous, as an instructor for one course in Fall A of 2010 suffered a stroke shortly after grade submission. The Department Chair was able to utilize the instructor's incomplete grade notes that were submitted during the grading process to work with one particular student to successfully complete the course.

Table 4. Incomplete Grades per Semester

Semester	**Incomplete Grades**
Fall 2009	1244
Spring 2010	835
Summer 2010	251
Fall A, 2010	91
Fall B, 2010	120
Fall 2010 (total)	723

One of the deliverables that produced surprising results was the introduction of the date requirement for the failure due to non-attendance system. This system was put in place due to a federal financial aid mandate, and required that instructors enter in a last date of attendance if the student failed due to non-attendance. Due to the stringent guidelines of this regulation, the Financial Aid and Academic Affairs departments co-hosted numerous training sessions for faculty members on how to properly determine if a failure was due to non-attendance and how to appropriately track and mark the student for such. The results can be seen in Table 5, which highlight failure and failure due to non-attendance grades across consecutive semesters. Please note that a special grade was created for use in Spring 2010 that did not exist in Fall 2009 for tracking failure due to non-attendance, hence the value of 0.

Table 5. F/FN Grades per Semester

Semester	**F Grades**	**FN Grades**
Fall 2009	9141	0
Spring 2010	6770	3418
Summer 2010	1820	800
Fall A, 2010	1139	533
Fall B, 2010	1054	922
Fall 2010 (total)	6385	5211

One can see that the introduction of the failure due to non-attendance system with a required date resulted in an overall reduction in failure grades. Furthermore, appropriate training by the Financial Aid and Academic Affairs departments kept failure due to non-attendance submissions well below total failure submissions. The introduction of the FN

grade for the Spring 2010 semester also reduced the total number of failure submissions that was exceedingly high during the Fall 2009 semester.

Another tangible result from the implementation of the system was the reduction of missing grade audit records. This particular phenomenon has been a constant source of audit criticisms against Daytona State College. The aforementioned blank white screen was a telltale symptom of this problem, and the numbers shown in Table 6 indicate just how widespread this problem reached.

Table 6. Missing Grade Audit Records per Semester

Semester	**Missing Records**
Fall 2009	2788
Spring 2010	37
Summer 2010	653
Fall A, 2010	0
Fall B, 2010	0
Fall 2010 (total)	0

While the numbers shown thus far may be staggering, it is worth noting just how many sections and grades are submitted per year. You can see in Table 7 that in the past decade, Daytona State College has nearly doubled the number of sections, and more than doubled the number of grades submitted. Worth noting is the years of 2007 and 2008. During this time, lectures and labs were combined into a single course, before Florida's Department of Education changed back to separated courses. This is what prompted the

request for the shared grading system, especially given the high number of sections and grades that are submitted each year.

Table 7. Sections and Grades per Year

Year	Sections	Grades
2000	4595	64943
2001	4654	69163
2002	4753	73629
2003	4864	78002
2004	5476	82395
2005	7353	83648
2006	7351	85916
2007	6634	99862
2008	6772	114860
2009	6918	136598
2010	7362	152817

Finally, one may wonder about the apparent discrepancy between Fall A and B totals versus totals for the entire Fall semester. Daytona State College offers many courses outside of the normal 8-week and 16-week semesters. This results in totals that appear to not accurately reflect a transition from the first half to second half of a semester.

CONCLUSIONS

The implementation of the new class final grading system appears to be an unqualified success. There has been a wholesale reduction in all of the metrics that showed the vulnerability of the old system, as well as a vast improvement in user experience. Furthermore, the system now meets or exceeds ADA and web standards, with a look and feel that more closely matches the design style the college is pursuing.

All requirements put forth by the users as well as technical constraints have been met, and the system has been in active, production use since the first grading cycle of Fall 2010. Several enhancements requests have also been processed, with implementations either already complete or planned for upcoming development cycles. For instance, a request was processed and completed at the beginning of the Spring 2011 semester to introduce color coding to student names on the class roster to indicate the student's academic status. This should allow faculty members to more easily identify students at risk of failing a class and further hurting their academic career.

Throughout the process, the initial design was tweaked to fit the needs of the system and the technical capabilities of technology already in place. Due to political constraints the timeline that was initially established for the project was altered substantially, and the planned testing phase with an initial group of faculty was abandoned in favor of full production use. With this last minute change, not all faculty were adequately informed of the change to the new system in time, and a flurry of confused calls were fielded by Academic Affairs before all faculty were fully informed and trained.

Since there was such a drastic change at the last minute that resulted in inadequately tested code being utilized, changes will be made to all further project plans to expand the developmental timetable to take political changes into account and thus allow adequate testing. Furthermore, it was discovered during the evaluation phase that various aspects of this system can be adapted to a variety of other systems that are also in need of extensive maintenance. The modular nature of the system's design will allow it to be introduced to further systems with little to no alterations.

Looking strictly at the class final grading system, there will be further changes integrated into it within the next year. First and foremost among these will be a language switch from JSP to C#. As DSC continues to move forward with advanced implementation of the Jenzabar Internet Campus Solution, or JICS Portal, all of the current systems written in JSP will be converted to C# equivalents. Beyond the language update, other user requests are currently pending completion according to the development schedule. A brief listing of these requests can be found in Table 7.

Table 8. Pending Class Final Grading Requests

Pending Requests
Implement support for last date of attendance for F grade.
Implement system to verify student has met requisites for the course they are in.
Provide access to a contact form for the student rather than just providing the email address.
Provide ability to sort student list by methods other than alphabetical by last name.

REFERENCES

Engrade. (2011). *Engrade*. Retrieved February 4, 2011, from Engrade: http://www.engrade.com/

SunGard Higher Education. (2011). *PowerCAMPUS Student*. Retrieved January 25, 2011, from SunGard Higher Education: http://www.sungardhe.com/Solutions.aspx?id=1268

Zimmerman, D. M., Kiniry, J. R., & Fairmichael, F. (2011). Toward Instant Gradeification. *24th IEEE-CS Conference on Software Engineering Education and Training* (pp. 1-5). IEEE.

APPENDIX A: USERS' MANUAL/HELP DOCUMENTATION

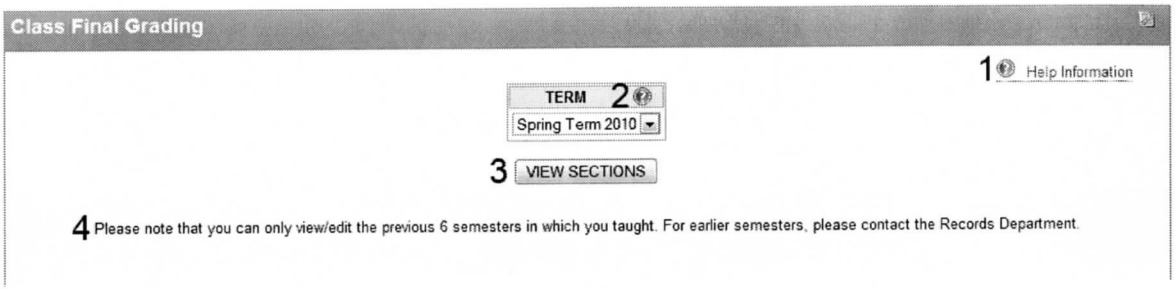

Figure 4. Term Selection Help

1. Full page help information (this document).
2. Context-sensitive help information. This will display help information about the specific topic.
3. You must click View Sections to continue.
4. The grading system will only display a limited range of terms in which you taught. Currently, this limit is 6. You will not be able to view terms in which you didn't teach. Thus, if you only teach in Fall and Spring, you will not see a Summer term listed.

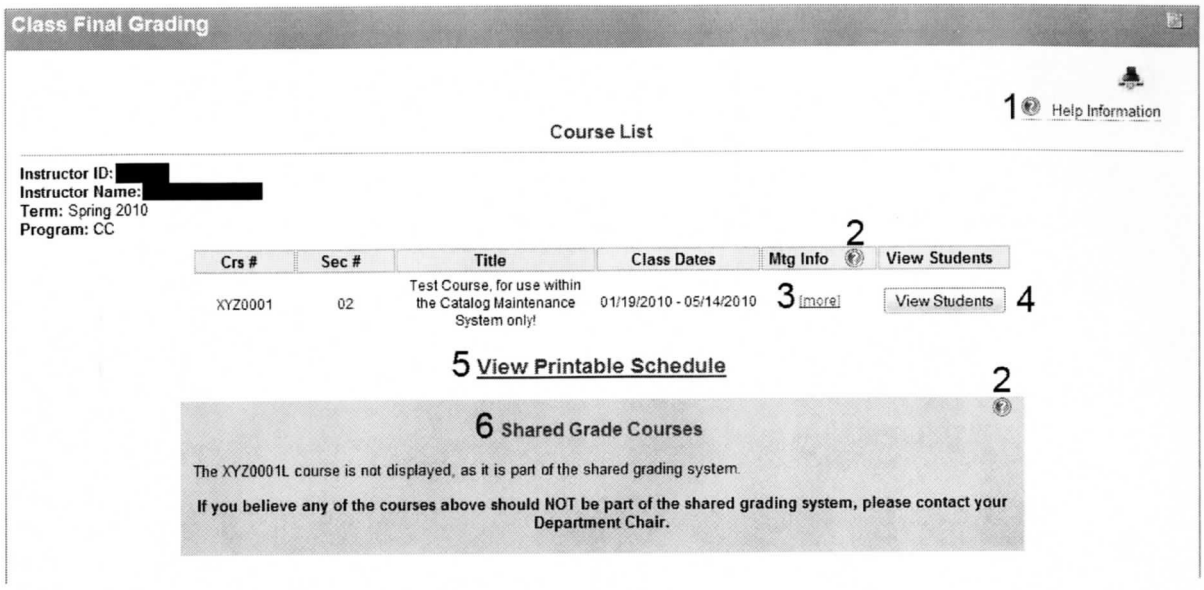

Figure 5. Course Selection Help

1. Full page help information (this document).
2. Context-sensitive help information. This will display help information about the specific topic.
3. This link contains more information about the section. This includes each meeting record and registered/enrolled numbers.
4. The View Students button will get you into your class list (roster).
5. The View Printable Schedule link will give you a printable version of your schedule that contains all of your meetings along with locations. The Printable Schedule page is specially-formatted to provide the best print capability for college printers.
6. The Shared Grade Courses box contains all of the courses you teach that are part of the Shared Grading system. You are not able to grade these courses, and thus, you are not able to view the class list (roster) for these courses.

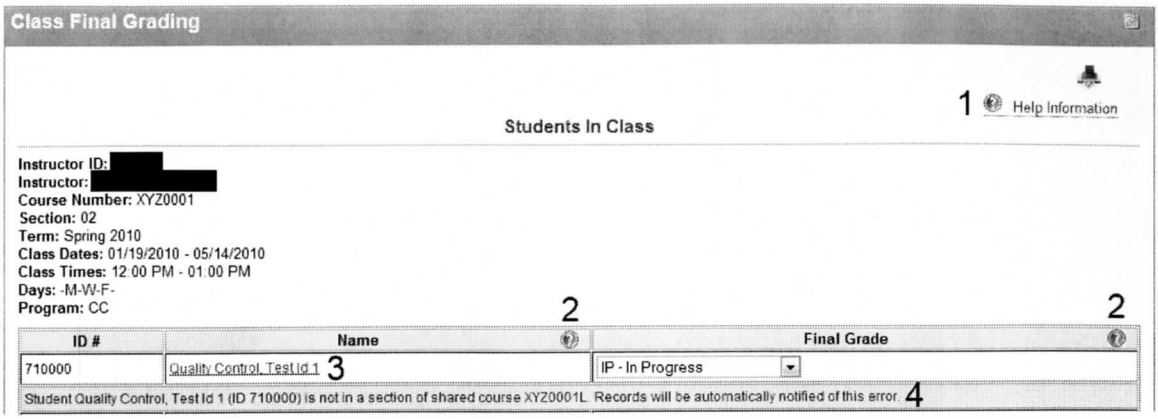

Figure 6. Class Roster Help, Screen 1

1. Full page help information (this document).
2. Context-sensitive help information. This will display help information about the specific topic.
3. This link contains more information about the student. Here you will find the student's phone number, FalconMail email address, and an alternate email address (if available). It is recommended that you NOT use the alternate email address for any sensitive information, such as grades.
4. If an error exists for a student, you will see it here. Any error that exists for a student will disable the grading system (to ensure accurate grading). All errors will be automatically emailed to the Records Department for resolution. It is your responsibility to verify the problem has been fixed by contacting Records.

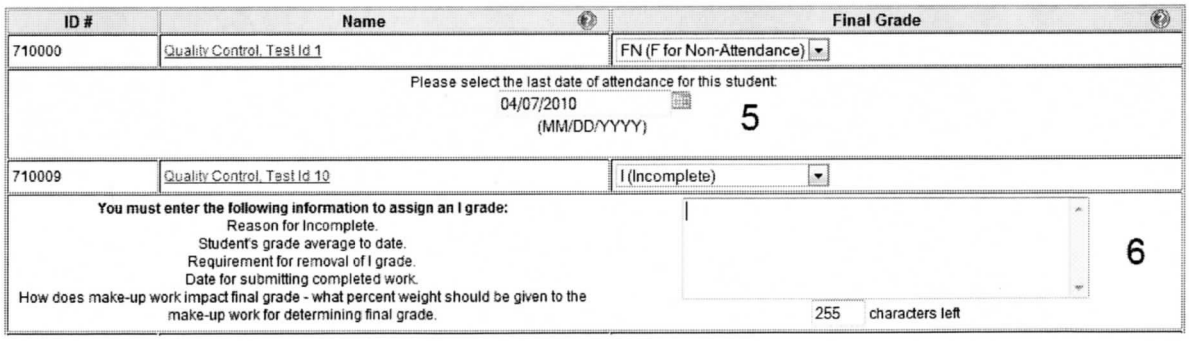

Figure 7. Class Roster Help, Screen 2

5. Any student receiving a F for Non-Attendance (FN) will require a last date of attendance. The calendar icon next to the date will allow you to use a graphical calendar to pick the date, or you can type the date in the format of MM/DD/YYYY. The range MUST be after the add/drop date and before the end of the section.
6. When you submit an Incomplete grade, you will need to provide a list of materials and other information in the textbox. This will be logged as a TRACS alert, and a copy will go to the Advisor (or, if the student has no Advisor, the main Advising Department email). In the near future, a copy will also go to the student.

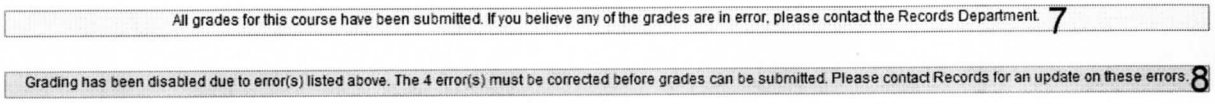

Figure 8. Class Roster Help, Screen 3

7. If you have already graded all of your students, the system will give you this notification and prevent you from submitting your grades again.
8. If you have any errors, grading will be disabled until the errors are fixed. You will see this notification stating such. Records is automatically notified of any errors encountered, but it is your responsibility to verify with Records that the problem has been resolved.

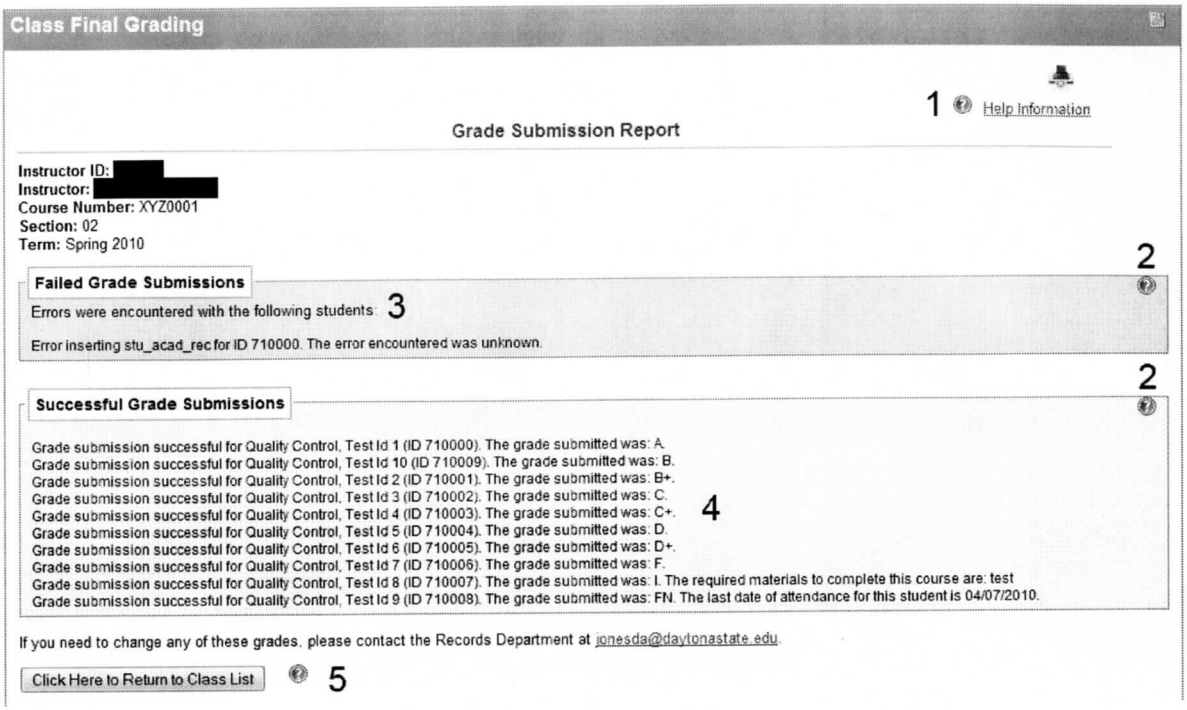

Figure 9. Grade Submission Help

1. Full page help information (this document).
2. Context-sensitive help information. This will display help information about the specific topic.
3. If there were any errors during submission, they will be displayed here. A copy of all of the errors will be emailed to the Records Department to have action taken. It is your responsibility to follow up with Records to ensure that the errors have been corrected.
4. A copy of all successful submissions will be shown on the screen for you to see. Additionally, you will be able to print this list out in a printer-friendly format. A copy of your successful submissions, as well as any errors, will be sent to your email address. If you have an Exchange account, the email will go to your Outlook email. Otherwise, the email will be sent to your FalconMail account. If you need to change any of these grades, you will need to contact the **Records Department**. The Help

Desk **CANNOT** change grades or submit the request for you, that is entirely your responsibility.

5. You may click on this button to return to your class list. Do **NOT** attempt to use your browser's Back button. If you use your browser's Back button, you may be unable to view your class list due to security measures put in place to ensure accurate reporting.

APPENDIX B: SYSTEM TECHNICAL DOCUMENTATION

TABLE SCHEMAS

aa_rec (Alternate Address Record)

```
CREATE
  TABLE aa_rec
  (
    id INTEGER DEFAULT 0 NOT NULL,
    aa CHAR(4) NOT NULL,
    beg_date DATE NOT NULL,
    end_date DATE,
    peren CHAR(1),
    line1 CHAR(32),
    line2 CHAR(32),
    line3 CHAR(32),
    city CHAR(32),
    st CHAR(2),
    zip CHAR(10),
    ctry CHAR(3),
    phone CHAR(12),
    phone_ext CHAR(4),
    ofc_add_by CHAR(4),
    cass_cert_date DATE,
    aa_no SERIAL NOT NULL
  )
```

acad_cal_rec (Academic Calendar Record)

```
CREATE
  TABLE acad_cal_rec
  (
    prog CHAR(4) NOT NULL,
    yr SMALLINT DEFAULT 0 NOT NULL,
    sess CHAR(4) NOT NULL,
```

```
subsess CHAR(2) NOT NULL,
subsess_grp CHAR(1) NOT NULL,
acyr CHAR(4),
beg_date DATE,
end_date DATE,
first_reg_date DATE,
last_add_date DATE,
last_drop_date DATE,
last_wd_date DATE,
last_wd2_date DATE,
last_reg_date DATE,
charge_date DATE,
rpt_date1 DATE,
rpt_date2 DATE,
weeks SMALLINT DEFAULT 0 NOT NULL,
ft_hrs SMALLINT DEFAULT 0 NOT NULL,
ft_clock_hrs SMALLINT DEFAULT 0 NOT NULL,
phrase_no SMALLINT,
last_grdg_date DATE,
schd_print_date DATE,
last_adm_date DATE,
first_midgrd_upd DATE,
last_midgrd_upd DATE,
first_grd_upd DATE,
last_grd_upd DATE,
first_adv_reg DATE,
last_adv_reg DATE,
first_stu_reg DATE,
last_stu_reg DATE,
open_enr_last_add DATE,
last_grd_upd_tm SMALLINT,
```

```
    midgrd_view_date DATE,
    grd_view_date DATE,
    web_display_date DATE,
    web_display_end DATE,
    first_incmpl_upd DATE,
    last_incmpl_upd DATE,
    max_hrs FLOAT,
    high_date DATE,
    low_date DATE,
    purge_date DATE,
    unprotect_fa_date DATE,
    last_purge_date DATE,
    igrd_change_date DATE,
    cat CHAR(4),
    data_final CHAR(1) NOT NULL,
    first_finals_date DATE,
    last_finals_date DATE,
    open_for_atc CHAR(1) NOT NULL
  )
```

acad_stat_table (Academic Status Table)

```
CREATE
  TABLE acad_stat_table
  (
    acst CHAR(4) NOT NULL,
    txt CHAR(24),
    reg CHAR(1),
    warn CHAR(1),
    adm_upd CHAR(1),
    cc_code CHAR(1),
    active_date DATE,
    inactive_date DATE
```

```
  )
```

addree_rec (Addressee Record)

```
CREATE
  TABLE addree_rec
  (
    addree_no SERIAL NOT NULL,
    prim_id INTEGER DEFAULT 0 NOT NULL,
    sec_id INTEGER DEFAULT 0 NOT NULL,
    active_date DATE,
    inactive_date DATE,
    addree CHAR(1) NOT NULL,
    style CHAR(1) NOT NULL,
    id_used_by INTEGER DEFAULT 0 NOT NULL,
    title CHAR(4),
    suffix CHAR(4),
    alt_name CHAR(32) NOT NULL,
    alt_ss_no CHAR(11) NOT NULL,
    line_sndx CHAR(4) NOT NULL,
    ofc_add_by CHAR(4)
  )
```

altcal_rec (Alternate Calendar Record)

```
CREATE
  TABLE altcal_rec
  (
    crs_no CHAR(12) NOT NULL,
    cat CHAR(4) NOT NULL,
    yr SMALLINT DEFAULT 0 NOT NULL,
    sess CHAR(4) NOT NULL,
    sec_no CHAR(2) NOT NULL,
    last_add_date DATE,
    last_drop_date DATE,
```

```
    last_wd_date DATE,
    charge_date DATE,
    last_grdg_date DATE,
    midgrd_view_date DATE,
    grd_view_date DATE,
    id INTEGER DEFAULT 0 NOT NULL
  )
```

crs_rec (Course Record)

```
CREATE
  TABLE crs_rec
  (
    crs_no CHAR(12) NOT NULL,
    cat CHAR(4) NOT NULL,
    cip_no CHAR(8),
    dept CHAR(4),
    prog CHAR(4),
    title1 CHAR(32),
    title2 CHAR(32),
    title3 CHAR(32),
    rep CHAR(1),
    max_rep SMALLINT,
    max_rep_hrs FLOAT,
    grd_rep CHAR(1),
    grdg CHAR(2),
    cntg CHAR(2),
    lvl CHAR(2),
    tuit_code CHAR(4),
    fee_code CHAR(4),
    bill_code CHAR(4),
    min_hrs FLOAT,
    max_hrs FLOAT,
```

```
clock_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
pref_cl_size SMALLINT,
vo_ed CHAR(1),
fac_consent CHAR(1),
fac_consent_no INTEGER,
indep_study_allow CHAR(1),
dir_study_allow CHAR(1),
core CHAR(4),
yr_offer SMALLINT,
sess_offer CHAR(4),
cr_rep SMALLINT,
cr_max_rep SMALLINT,
max_days SMALLINT,
last_drop_days SMALLINT,
site CHAR(4),
stat CHAR(1),
stat_date DATE,
accept_date DATE,
fac_load CHAR(1),
phrase_no SMALLINT,
grdg_disallow CHAR(2) NOT NULL,
disc CHAR(4),
crsctgry CHAR(4),
degapply CHAR(2),
prim_assoc CHAR(4),
schedsort CHAR(4),
gordon INTEGER,
fund CHAR(2),
subfund CHAR(9),
FUNCTION CHAR(5),
coop_ed CHAR(4),
```

```
    rpt_major CHAR(4) NOT NULL,
    db_fc_rep SMALLINT DEFAULT 0 NOT NULL,
    sectitle_upd CHAR(1),
    prim_lcp CHAR(5) NOT NULL,
    lecture_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    lab_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    db_curr_crs_no CHAR(12) NOT NULL,
    allow_over_reg CHAR(1) NOT NULL
  )
```

crsauth_rec (Course Authorization Record)

```
CREATE
  TABLE crsauth_rec
  (
    crsauth_no SERIAL NOT NULL,
    id INTEGER DEFAULT 0 NOT NULL,
    crs_no CHAR(12) NOT NULL,
    cat CHAR(4) NOT NULL,
    yr SMALLINT DEFAULT 0 NOT NULL,
    sess CHAR(4) NOT NULL,
    sec_no CHAR(2) NOT NULL,
    reason CHAR(2),
    condition CHAR(8),
    upd_date DATE,
    upd_uid SMALLINT,
    upd_user_id INTEGER,
    stat CHAR(1) NOT NULL,
    end_date DATE
  )
```

cw_rec (Coursework Record)

```
CREATE
  TABLE cw_rec
```

```
(
    cw_no SERIAL NOT NULL,
    id INTEGER DEFAULT 0 NOT NULL,
    prog CHAR(4) NOT NULL,
    yr SMALLINT DEFAULT 0 NOT NULL,
    sess CHAR(4) NOT NULL,
    subsess CHAR(2) NOT NULL,
    crs_no CHAR(12) NOT NULL,
    cat CHAR(4) NOT NULL,
    sec CHAR(2) NOT NULL,
    hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    clock_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    grd CHAR(3),
    midsess_grd CHAR(3),
    rep CHAR(2),
    grdg CHAR(2),
    cntg CHAR(2),
    beg_date DATE,
    end_date DATE,
    stat CHAR(1),
    absnt_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    absnt_upd CHAR(1),
    site CHAR(4),
    rawgrd CHAR(3),
    midsess_rawgrd CHAR(3),
    regctgry CHAR(8),
    last_attend_date DATE,
    paid CHAR(1),
    db_fc_flag CHAR(1) NOT NULL,
    db_units FLOAT DEFAULT 0.0000000000000000 NOT NULL
)
```

db_crs2_rec (Supplemental Course Record)

```
CREATE
  TABLE db_crs2_rec
  (
    crs_no CHAR(12) NOT NULL,
    cat CHAR(4) NOT NULL,
    carnegie FLOAT DEFAULT 0.5000000000000000 NOT NULL,
    prog CHAR(2) NOT NULL,
    lecture_ratio FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    lecture_cred_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    lab_ratio FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    lab_cred_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    clinical_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    fee_code1 CHAR(4) NOT NULL,
    fee_code2 CHAR(4) NOT NULL,
    fee_code3 CHAR(4) NOT NULL,
    true_master CHAR(1) NOT NULL,
    load_credits FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    altload_credits1 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    altload_credits2 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    pay_hrs SMALLINT DEFAULT 0 NOT NULL,
    title CHAR(90) NOT NULL,
    altload_credits3 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    altload_credits4 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    allow_cross_sems CHAR(1) NOT NULL,
    alt_instr_load1 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    alt_instr_load2 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    load_factor FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    alt_instr_pay_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    allow_open_enroll CHAR(1) NOT NULL,
    alt_instr_load3 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
```

```
    alt_instr_load4 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    alt_instr_payhrs2 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    alt_instr_payhrs3 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    alt_instr_payhrs4 FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    min_virtual_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    max_virtual_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    share_grade CHAR(12) NOT NULL,
    allow_mult_reg CHAR(1) DEFAULT 'N',
    prog_guide_crs CHAR(1) NOT NULL,
    fld_code CHAR(4) NOT NULL
  )
```

db_cw2_rec (Supplemental Coursework Record)

```
CREATE
  TABLE db_cw2_rec
  (
    cw_no INTEGER NOT NULL,
    paid_date CHAR(23),
    fgrade_attend CHAR(1),
    faa_ver CHAR(1),
    attendver_date DATE,
    fac_id INTEGER DEFAULT 0 NOT NULL,
    req_met CHAR(1),
    auth_met CHAR(1),
    comment_code CHAR(4)
  )
```

db_email_rec (FalconMail Email Record)

```
CREATE
  TABLE db_email_rec
  (
    id INTEGER,
    network_login CHAR(8),
```

```
userid VARCHAR(40),
fullname VARCHAR(40),
password CHAR(6) DEFAULT 'yy4577',
userdir VARCHAR(70) DEFAULT 'e:\email\users\default',
add_date DATE,
permanent CHAR(1),
cars_stat CHAR(1),
network_stat CHAR(1),
email_stat CHAR(1),
exp_date DATE,
mailaddr VARCHAR(255),
maxsize INTEGER DEFAULT 0,
maxmsgs INTEGER DEFAULT 0,
flags INTEGER DEFAULT 4230,
type INTEGER DEFAULT 40968,
numtries INTEGER DEFAULT 0,
attempttime INTEGER DEFAULT 0,
numtimessusp INTEGER DEFAULT 0,
lastlogin INTEGER DEFAULT 0,
spndusracct INTEGER DEFAULT 0,
iwoptions INTEGER DEFAULT 0,
spellchkg INTEGER DEFAULT 0,
icalenbl INTEGER DEFAULT 1,
spellenbl INTEGER DEFAULT 0,
numsgstns INTEGER DEFAULT 0,
iwdepth INTEGER DEFAULT 0,
prvwsize INTEGER DEFAULT 0,
strtday INTEGER DEFAULT 0,
entrygrnl INTEGER DEFAULT 60,
starttime INTEGER DEFAULT 8,
endtime INTEGER DEFAULT 17,
```

```
    maxevntdisp INTEGER DEFAULT 0,
    maxtskdisp INTEGER DEFAULT 0,
    maxnotedisp INTEGER DEFAULT 0,
    maxdlyrcurcnt INTEGER DEFAULT 365,
    maxwklyrcurcnt INTEGER DEFAULT 100,
    maxmonrcurcnt INTEGER DEFAULT 36,
    maxyrlyrcurcnt INTEGER DEFAULT 0,
    dispcmpltdtsks INTEGER DEFAULT 0,
    hoursclock INTEGER DEFAULT 12,
    maxsrchresppg INTEGER DEFAULT 10,
    maxeventtitlelen INTEGER DEFAULT 50,
    maxtasktitlelen INTEGER DEFAULT 10,
    confirmondelete INTEGER DEFAULT 0,
    numnotify INTEGER DEFAULT 0,
    lastnotify INTEGER DEFAULT 0,
    enablealtwgsid INTEGER DEFAULT 0,
    altwgsid INTEGER DEFAULT 0,
    timezone VARCHAR(255) DEFAULT 'D Eastern Standard Time',
    defview VARCHAR(40) DEFAULT 'MONTHLY',
    defcalnm VARCHAR(40) DEFAULT 'Personal',
    trig_flag CHAR(1) DEFAULT 'A'
  )
```

db_webdocs_table (Web Documents Table)

```
CREATE
  TABLE db_webdocs_table
  (
    uid SERIAL NOT NULL,
    module CHAR(20),
    doc TEXT,
    descr CHAR(50),
    active_date DATE,
```

```
    inactive_date DATE
  )
```

dbcampus_rec (Supplemental Campus Record)

```
CREATE
  TABLE dbcampus_rec
  (
    campus_num CHAR(4),
    campus_name CHAR(16),
    valid_camp CHAR(1),
    section_display CHAR(1),
    adv_priority SMALLINT,
    merfnbr SMALLINT
  )
```

dbtracs_rec (TRACS Record)

```
CREATE
  TABLE dbtracs_rec
  (
    uid SERIAL NOT NULL,
    stu_id INTEGER,
    yr INTEGER,
    sess CHAR(4),
    crs_no CHAR(12),
    sec_no CHAR(2),
    instr_id INTEGER,
    stat VARCHAR(10),
    COMMENT VARCHAR(255),
    adv_actions VARCHAR(255),
    active_date DATE,
    update_date DATE,
    inactive_date DATE
  )
```

dept_table (Department Table)

```
CREATE
  TABLE dept_table
  (
    dept CHAR(4) NOT NULL,
    txt CHAR(24),
    div CHAR(4) NOT NULL,
    active_date DATE,
    inactive_date DATE,
    web_display CHAR(1),
    chair_id INTEGER DEFAULT 0
  )
```

grdau_rec (Grade Audit Record)

```
CREATE
  TABLE grdau_rec
  (
    grdau_no SERIAL NOT NULL,
    cw_no INTEGER DEFAULT 0 NOT NULL,
    cntg CHAR(2),
    grd CHAR(3),
    grdg CHAR(2),
    hrs FLOAT,
    uid SMALLINT,
    user_id INTEGER,
    upd_date DATE,
    upd_tm INTEGER,
    inform_date DATE,
    new_cntg CHAR(2),
    new_grd CHAR(3),
    new_grdg CHAR(2),
    new_hrs FLOAT,
```

```
    rep_date DATE,
    mass_chg CHAR(1) NOT NULL
  )
```

id_rec (ID Record)

```
CREATE
  TABLE id_rec
  (
    id SERIAL NOT NULL,
    prsp_no INTEGER DEFAULT 0 NOT NULL,
    fullname CHAR(32) NOT NULL,
    name_sndx CHAR(4) NOT NULL,
    lastname CHAR(50) NOT NULL,
    firstname CHAR(32) NOT NULL,
    middlename CHAR(32) NOT NULL,
    suffixname CHAR(10) NOT NULL,
    addr_line1 CHAR(24),
    addr_line2 CHAR(24),
    addr_line3 CHAR(24),
    city CHAR(24),
    st CHAR(2),
    zip CHAR(10) NOT NULL,
    ctry CHAR(3),
    aa CHAR(4),
    title CHAR(4),
    suffix CHAR(4),
    ss_no CHAR(11) NOT NULL,
    phone CHAR(12),
    phone_ext CHAR(4),
    prev_name_id INTEGER,
    mail CHAR(1),
    sol CHAR(1),
```

```
    pub CHAR(1),
    correct_addr CHAR(1),
    decsd CHAR(1),
    add_date DATE,
    ofc_add_by CHAR(4),
    upd_date DATE,
    valid CHAR(1),
    purge_date DATE,
    cass_cert_date DATE,
    uid_add_by SMALLINT DEFAULT 0 NOT NULL,
    uid_upd_by SMALLINT DEFAULT 0 NOT NULL,
    PRIMARY KEY (id) CONSTRAINT id_id
  )
```

im_table (Instructional Method Table)

```
CREATE
  TABLE im_table
  (
    im CHAR(2) NOT NULL,
    descr CHAR(24),
    flt CHAR(2),
    aam CHAR(2),
    cc_code CHAR(2),
    web_display CHAR(1) DEFAULT 'Y',
    active_date DATE,
    inactive_date DATE,
    virtual FLOAT DEFAULT 0.0000000000000000,
    virtual_flag CHAR(1) DEFAULT 'N',
    use_alt_fte CHAR(1) DEFAULT 'N',
    db_im_ctgry CHAR(2) NOT NULL
  )
```

instr_rec (Instructor Record)

```
CREATE
  TABLE instr_rec
  (
    instr_no SERIAL NOT NULL,
    mtg_no INTEGER DEFAULT 0 NOT NULL,
    yr SMALLINT DEFAULT 0 NOT NULL,
    sess CHAR(4) NOT NULL,
    id INTEGER DEFAULT 0 NOT NULL,
    flt CHAR(2),
    accrl_meth CHAR(1),
    fte FLOAT,
    fte_man FLOAT,
    pct FLOAT,
    rpt CHAR(1),
    obj CHAR(5) NOT NULL,
    ppe_pct FLOAT,
    cntr_hrs FLOAT
  )
```

mtg_rec (Meeting Record)

```
CREATE
  TABLE mtg_rec
  (
    mtg_no SERIAL NOT NULL,
    yr SMALLINT DEFAULT 0 NOT NULL,
    sess CHAR(4) NOT NULL,
    campus CHAR(4) NOT NULL,
    bldg CHAR(4) NOT NULL,
    room CHAR(4) NOT NULL,
    beg_tm SMALLINT,
    end_tm SMALLINT,
    beg_date DATE,
```

```
    end_date DATE,
    days CHAR(7),
    hrs FLOAT,
    calc_tot_hrs FLOAT,
    tot_hrs FLOAT,
    im CHAR(2),
    reserve CHAR(1),
    schd_print CHAR(1),
    stat CHAR(1),
    s25_export CHAR(1) DEFAULT 'N',
    s25_assign CHAR(1) DEFAULT 'N',
    prd CHAR(2),
    schedule_str CHAR(8) NOT NULL,
    target_load CHAR(8) NOT NULL
  )
```

sec_rec (Section Record)

```
CREATE
  TABLE sec_rec
  (
    crs_no CHAR(12) NOT NULL,
    cat CHAR(4) NOT NULL,
    yr SMALLINT DEFAULT 0 NOT NULL,
    sess CHAR(4) NOT NULL,
    sec_no CHAR(2) NOT NULL,
    hrs FLOAT,
    clock_hrs FLOAT,
    subsess CHAR(2) NOT NULL,
    title CHAR(32),
    rstr CHAR(4),
    tuit_code CHAR(4),
    fee_code CHAR(4),
```

```
bill_code CHAR(4),
grdg CHAR(2),
fac_id INTEGER,
max_reg SMALLINT,
max_wait SMALLINT,
reg_num SMALLINT,
resrv_num SMALLINT,
enr_num SMALLINT,
wait_num SMALLINT,
exam_code CHAR(4),
beg_date DATE,
end_date DATE,
max_days SMALLINT,
remedial CHAR(1),
credit CHAR(1),
stat CHAR(1),
stat_date DATE,
ref_no INTEGER DEFAULT 0 NOT NULL,
class_list CHAR(1),
brdg_ord SMALLINT,
exch CHAR(1) DEFAULT 'W',
print_schd CHAR(1),
over_reg INTEGER,
altcal CHAR(2),
altrfnd_no INTEGER,
altrfnd CHAR(1),
open_enr CHAR(1),
weeks SMALLINT,
phrase_no SMALLINT,
grdg_disallow CHAR(2) NOT NULL,
schd_chg CHAR(1),
```

```
    max_reg_rsv SMALLINT DEFAULT 0 NOT NULL,
    reg_num_rsv SMALLINT DEFAULT 0 NOT NULL,
    deggrp CHAR(8) NOT NULL,
    pubreg CHAR(1) DEFAULT 'N',
    billing_date DATE,
    schd_upd_date DATE,
    fte_factor FLOAT,
    sec_locator CHAR(4) NOT NULL,
    db_type CHAR(2) NOT NULL,
    dbsec_type CHAR(2) NOT NULL,
    inst_flag CHAR(2) NOT NULL,
    fa_verified CHAR(1) NOT NULL,
    subprog CHAR(4)
  )
```

secmtg_rec (Section/Meeting Link Record)

```
CREATE
  TABLE secmtg_rec
  (
    crs_no CHAR(12) NOT NULL,
    cat CHAR(4) NOT NULL,
    yr SMALLINT DEFAULT 0 NOT NULL,
    sess CHAR(4) NOT NULL,
    sec_no CHAR(2) NOT NULL,
    mtg_no INTEGER DEFAULT 0 NOT NULL
  )
```

sess_table (Session Table)

```
CREATE
  TABLE sess_table
  (
    sess CHAR(4) NOT NULL,
    txt CHAR(24),
```

```
    yr_offset SMALLINT,
    ord SMALLINT,
    crs_tuit CHAR(1),
    is_prim CHAR(1),
    cc_code CHAR(1),
    active_date DATE,
    inactive_date DATE,
    web_display CHAR(1) DEFAULT 'Y',
    web_ord SMALLINT DEFAULT 0,
    trantxt CHAR(24),
    active CHAR(1)
  )
```

stu_acad_rec (Student Academic Record)

```
CREATE
  TABLE stu_acad_rec
  (
    id INTEGER DEFAULT 0 NOT NULL,
    prog CHAR(4) NOT NULL,
    site CHAR(4) NOT NULL,
    subprog CHAR(4),
    sess CHAR(4) NOT NULL,
    yr SMALLINT DEFAULT 0 NOT NULL,
    intend_hrs FLOAT,
    wait_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    reg_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    reg_au_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    att_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    earn_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    pass_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    qual_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
    au_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
```

```
qual_pts FLOAT DEFAULT 0.0000000000000000 NOT NULL,
gpa FLOAT DEFAULT 0.0000000000000000 NOT NULL,
crs_fees CHAR(1),
non_prog_crs CHAR(1),
grd_stat CHAR(1),
reg_stat CHAR(1),
fin_clr CHAR(1),
reg_upd_date DATE,
no_adrp SMALLINT,
cl_rank SMALLINT,
cl_size SMALLINT,
cl CHAR(2),
blk_code CHAR(4),
rnd INTEGER,
rpt_ipeds CHAR(1),
ipeds_hrs FLOAT,
sess_ord SMALLINT DEFAULT 0 NOT NULL,
acst CHAR(4),
goal CHAR(4),
wd_code CHAR(2),
wd_date DATE,
wd_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
last_attended_date DATE,
apprent CHAR(1),
voc_ed CHAR(1),
jtpa_stat CHAR(1),
gain_stat CHAR(1),
pbs_waiver CHAR(1),
res_st CHAR(2),
ofcl_trans_num SMALLINT,
ofcl_trans_chg SMALLINT,
```

```
unofcl_trans_num SMALLINT,
unofcl_trans_chg SMALLINT,
cum_att_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
cum_earn_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
cum_pass_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
cum_qual_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
cum_au_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
cum_qual_pts FLOAT DEFAULT 0.0000000000000000 NOT NULL,
cum_gpa FLOAT DEFAULT 0.0000000000000000 NOT NULL,
trnsfr_att_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
trnsfr_earn_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
trnsfr_pass_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
trnsfr_qual_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
trnsfr_au_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
trnsfr_qual_pts FLOAT DEFAULT 0.0000000000000000 NOT NULL,
trnsfr_gpa FLOAT DEFAULT 0.0000000000000000 NOT NULL,
res_att_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
res_earn_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
res_pass_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
res_qual_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
res_au_hrs FLOAT DEFAULT 0.0000000000000000 NOT NULL,
res_qual_pts FLOAT DEFAULT 0.0000000000000000 NOT NULL,
res_gpa FLOAT DEFAULT 0.0000000000000000 NOT NULL,
cl_end CHAR(2),
deggrp CHAR(8),
major1 CHAR(4),
major2 CHAR(4),
restart CHAR(1),
atb CHAR(1),
nhs CHAR(1),
voc_hrs FLOAT,
```

```
    currenr CHAR(4),
    incar_stat CHAR(4),
    wages_stat CHAR(4),
    col_prp_cmp CHAR(4),
    clock_hrs_att FLOAT,
    clock_hrs_earn FLOAT,
    award_crd_earn FLOAT,
    award_clock_earn FLOAT,
    entrtype CHAR(4),
    citz CHAR(4),
    db_due_date DATE
  )
```

userid_table (User ID Table)

```
CREATE
  TABLE userid_table
  (
    uid SMALLINT DEFAULT 0 NOT NULL,
    id_no INTEGER DEFAULT 0 NOT NULL,
    user_name CHAR(32),
    active_date DATE,
    inactive_date DATE,
    lastaccess_dt DATE,
    ofc CHAR(4),
    primary_group CHAR(15),
    default_prntrid CHAR(15),
    default_prntrgrp CHAR(15),
    entry_point CHAR(50),
    user_level INTEGER,
    operator_nm CHAR(24),
    cleanup CHAR(1)
  )
```

TABLE VIEWS

idnames_vw (ID/Demographic Information Link View)

```
CREATE VIEW
   idnames_vw ( id, fullname, lastname, firstname, midname, suffixname, greetingname, csz,
ssn ) AS
SELECT
   x0.id ,
   x0.fullname ,
   "informix".names_proc(x0.id ,'L' ),
   "informix".names_proc(x0.id ,'F' ),
   SUBSTR ("informix".names_proc(x0.id ,'M' ),1 ,1 ),
   "informix".names_proc(x0.id ,'S' ),
   "informix".names_proc(x0.id ,'G' ),
   ((((TRIM ( BOTH ' '
FROM
   x0.city ) || ', ' ) || x0.st ) || ' ' ) || x0.zip ) ,
   ((x0.ss_no [1,3] || x0.ss_no [5,6] ) || x0.ss_no [8,11] )
FROM
   "informix".id_rec x0 ;
```

PROCEDURES

names_proc (Names Procedure)

```
CREATE PROCEDURE informix.names_proc (
    idin integer          -- ID of the entity
   ,name_code char(1)          -- L-last name, F-first name, M-middle name
                        -- S-Suffix name, G-greeting name, W-whole name(fullname)
   ,in_fullname char(30) default " "  -- Can process the input (instead of retrieving from id..
not required
   )
  returning
```

```
    char(32);
define global old_id integer default 999999999;
define global wholename char(32) default " ";
define global lastname char(32) default " ";
define global firstname char(32) default " ";
define global midname char(32) default " ";
define global suffixname char(32) default " ";
define global greetingname char(32) default " ";
define thename char(32);
define n_index, beginname integer;
Set isolation to dirty read;
--set debug file to "debugddd" with append;
--trace on;
--If id has changed get new names else skip to the returns
if (idin <> old_id) then
let old_id = idin;
LET wholename = in_fullname;
IF (in_fullname = " ") THEN
 select fullname into wholename from id_rec where id = idin;
END IF
let lastname="x";
let firstname="x";
let midname="x";
let suffixname="x";
let beginname = 0;
let n_index = 0;
while (n_index < 32)
let n_index = n_index+1;
--Find lastname
if (lastname = "x") and (substr(wholename, n_index, 2) = ", ")  then
  let lastname = substr(wholename, 1, n_index-1);
```

```
  let beginname=n_index+2;
  let n_index=n_index+2;
end if;
if (n_index = 32) and (lastname = "x") then
  let lastname = wholename;
end if;
--Find firstname
if (lastname <> "x") and (firstname="x") and
  ((substr(wholename, n_index, 1) = " ") or
  (substr(wholename, n_index, 2) = ",,")) then
  let firstname = substr(wholename, beginname, n_index-beginname);
  let beginname=n_index+1;
end if;
if (n_index = 32) and (firstname = "x") and (lastname <> "x") then
  let firstname = substr(wholename, beginname, 32-beginname);
end if;
--Find midname and suffixname
if (firstname <> "x") and (midname="x") and
  (substr(wholename, n_index, 2) = ",,") then
  let midname = substr(wholename, beginname, 1);
  let suffixname = substr(wholename, n_index+2, 33-n_index+2);
  let n_index = 32;
end if;
if (n_index = 32) and (midname = "x") and (firstname <> "x") then
  let midname = substr(wholename, beginname, 1);
end if;
end while;
select min(alt_name) into greetingname from addree_rec
where addree_rec.prim_id = idin
and addree_rec.addree = "S"
and addree_rec.style = "I"
```

```
and addree_rec.id_used_by < 1
and addree_rec.inactive_date is NULL;
if (lastname="x") then
 let lastname=wholename;
end if;
if (firstname="x") then
 let firstname=" ";
end if;
if (midname="x") then
 let midname=" ";
end if;
if (suffixname="x") then
 let suffixname=" ";
end if;
if (greetingname is null) then
 let greetingname=firstname;
end if;
end if;
-- Target point of skip when id does not change.
if name_code = "L" then
 let thename = trim(lastname);
end if;
if name_code = "F" then
 let thename = trim(firstname);
end if;
if name_code = "M" then
 let thename = trim(midname);
end if;
if name_code = "S" then
 let thename = trim(suffixname);
end if;
```

```
if name_code = "G" then
 let thename = trim(greetingname);
end if;
if name_code = "W" then
 let thename = trim(wholename);
end if;
return thename;
end procedure;
```

APPENDIX C: WORK BREAKDOWN STRUCTURE

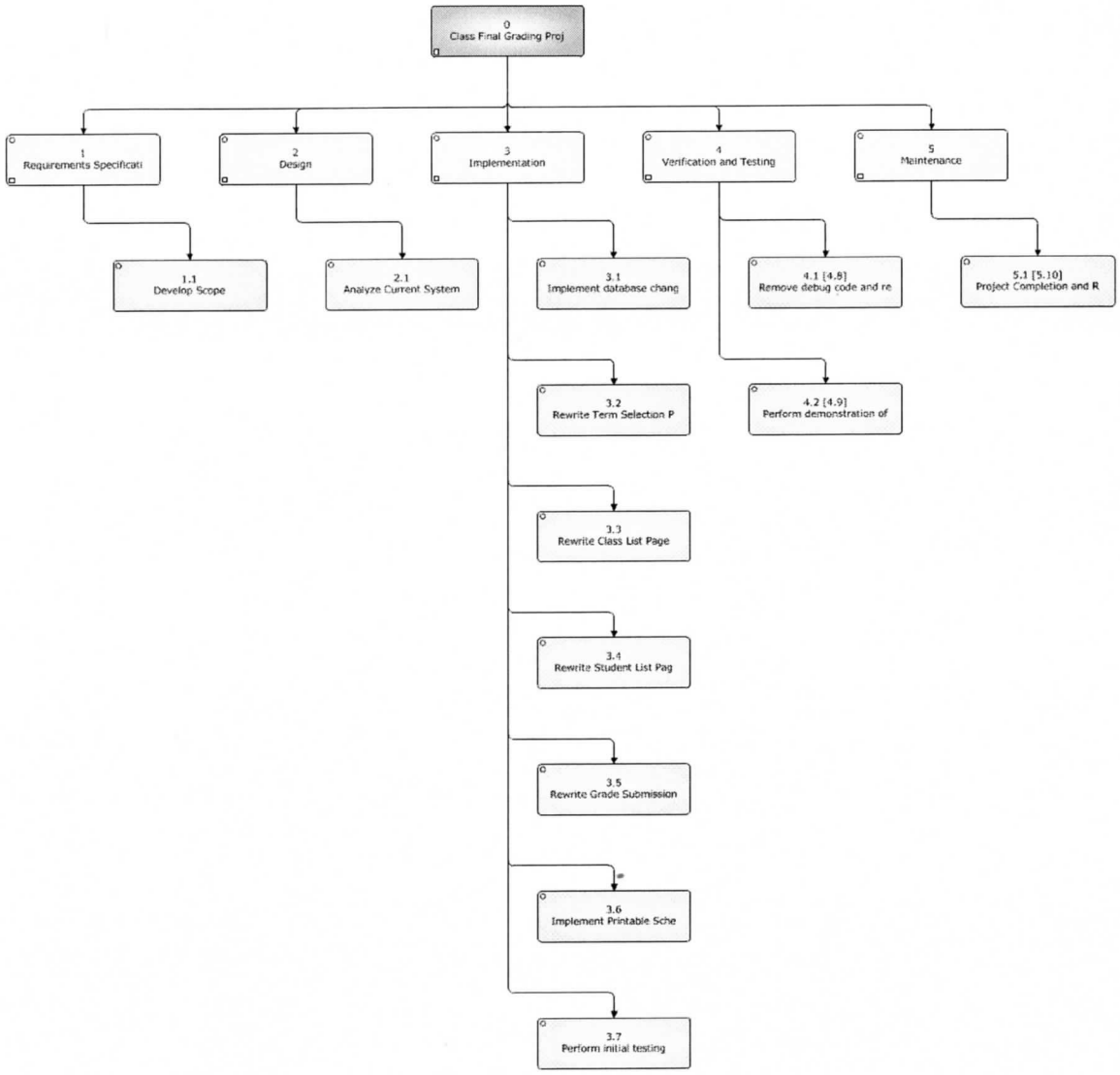

Figure 10. Work Breakdown Structure

APPENDIX D: GANTT CHART

ID	Task Name	Duration	Start	Finish
1	1 Requirements Specification	3 days	Mon 8/2/10	Wed 8/4/10
2	1.1 Develop Scope	3 days	Mon 8/2/10	Wed 8/4/10
7	2 Design	12 days	Thu 8/5/10	Fri 8/20/10
8	2.1 Analyze Current System	12 days	Thu 8/5/10	Fri 8/20/10
14	3 Implementation	20 days	Mon 8/23/10	Fri 9/17/10
15	3.1 Implement database changes	0.13 days	Mon 8/23/10	Mon 8/23/10
21	3.2 Rewrite Term Selection Page	0.13 days	Tue 8/24/10	Tue 8/24/10
29	3.3 Rewrite Class List Page	0.13 days	Wed 8/25/10	Wed 8/25/10
44	3.4 Rewrite Student List Page	2.25 days	Thu 8/26/10	Mon 8/30/10
96	3.5 Rewrite Grade Submission Page	2.06 days	Tue 8/31/10	Thu 9/2/10
135	3.6 Implement Printable Schedule	0.25 days	Thu 9/2/10	Thu 9/2/10
144	3.7 Perform initial testing using test instructors/courses	11 days	Fri 9/3/10	Fri 9/17/10
151	4 Verification and Testing	20.25 days	Mon 9/20/10	Mon 10/18/10
152	4.1 Remove debug code and release for beta testing by users	12 days	Mon 9/20/10	Tue 10/5/10
159	4.2 Perform demonstration of system for departments chairs and academic VPs	8.25 days	Wed 10/6/10	Mon 10/18/10
165	5 Maintenance	14.38 days	Mon 10/18/10	Fri 11/5/10
166	5.1 Project Completion and Release	14.38 days	Mon 10/18/10	Fri 11/5/10

Project: Class Final Grading Project P
Date: Fri 4/8/11

Task
Split
Milestone
Summary
Project Summary
External Tasks
External Milestone
Inactive Task
Inactive Milestone
Inactive Summary
Manual Task
Duration-only
Manual Summary Rollup
Manual Summary
Start-only
Finish-only
Progress
Deadline

Figure 11. Gantt Chart, Page 1

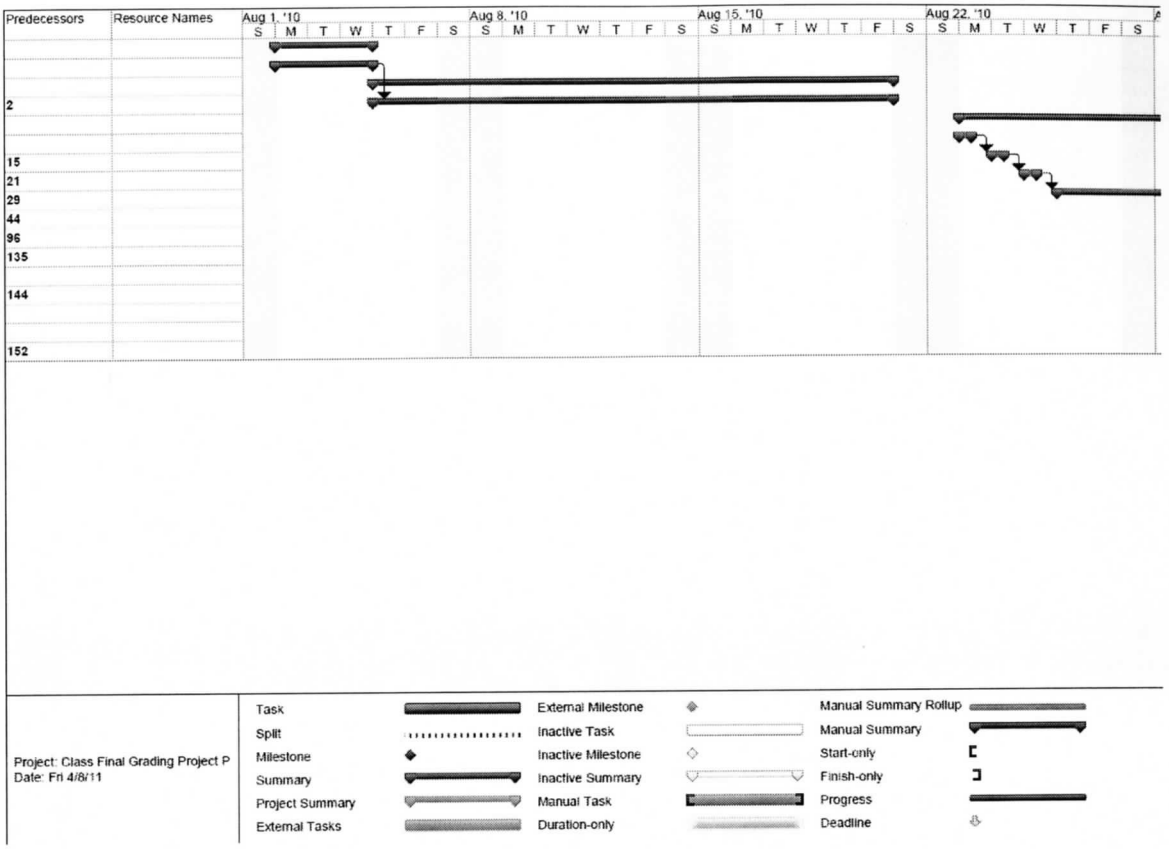

Figure 12. Gantt Chart, Page 2

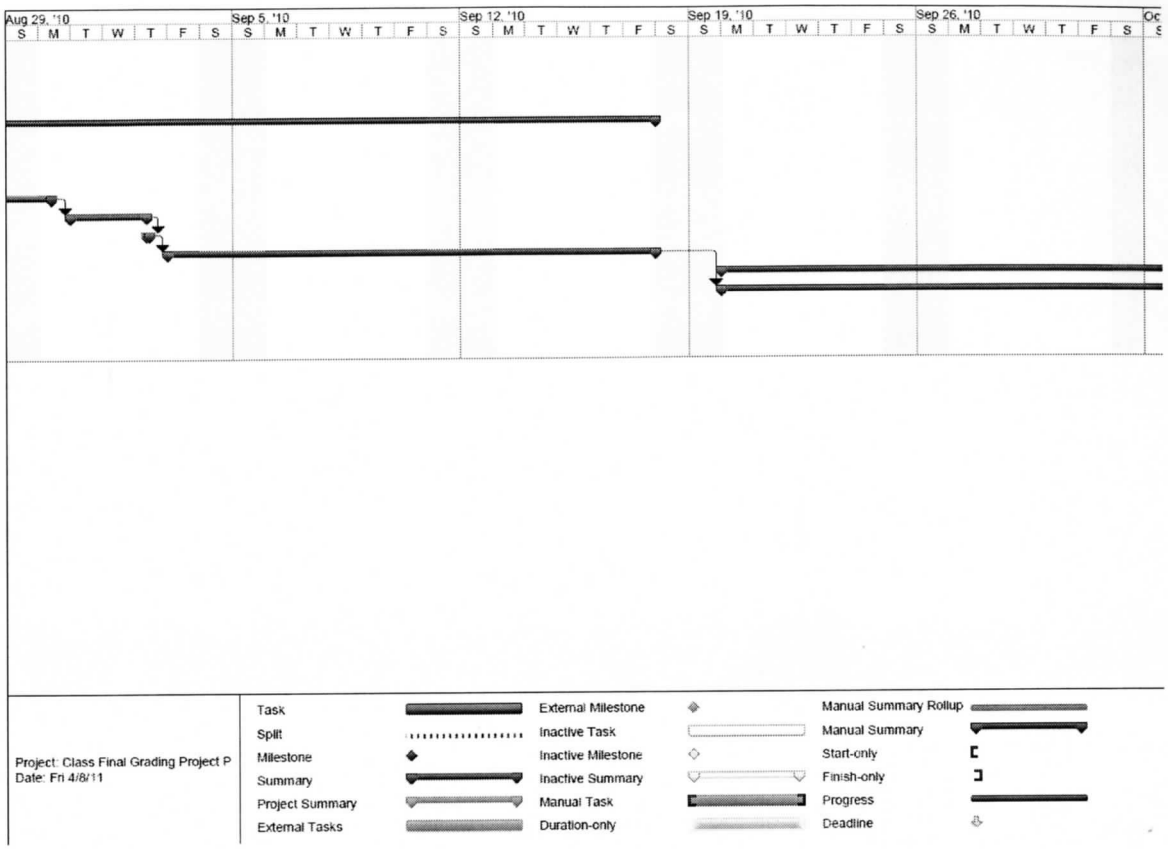

Figure 13. Gantt Chart, Page 3

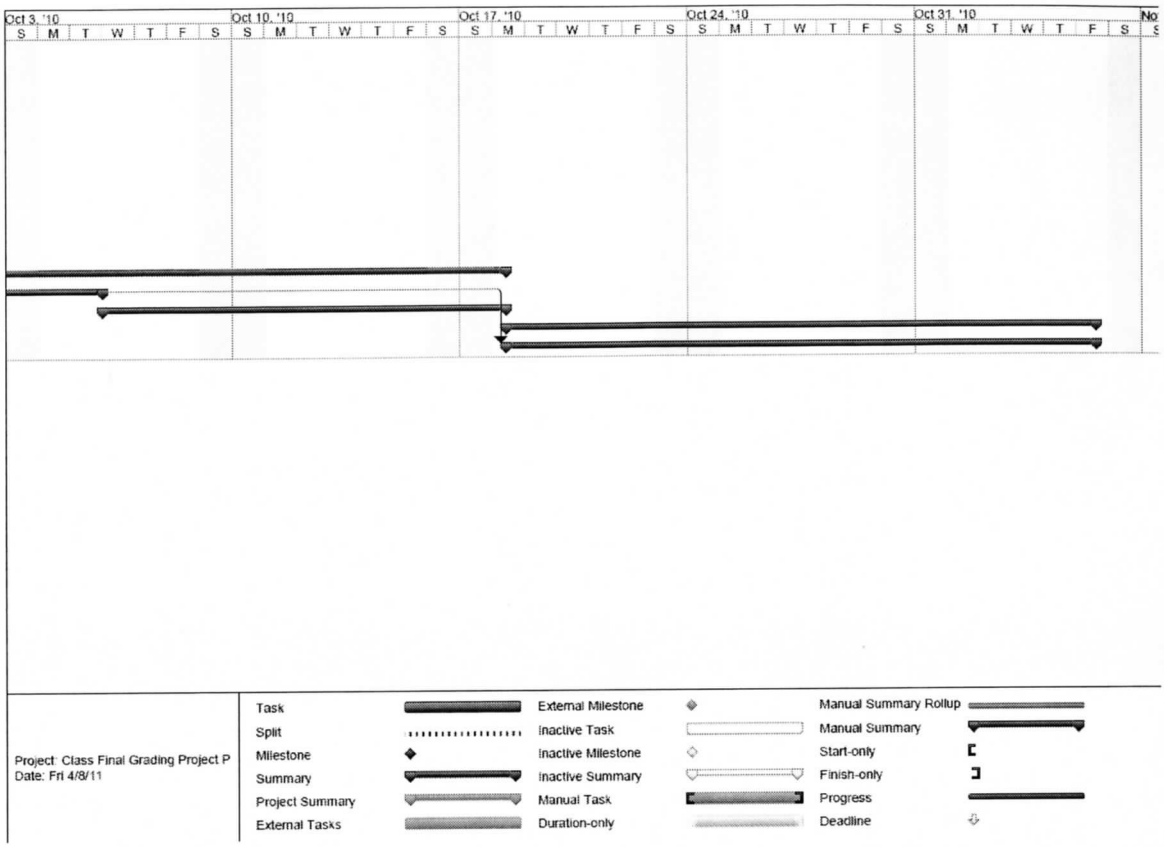

Figure 14. Gantt Chart, Page 4